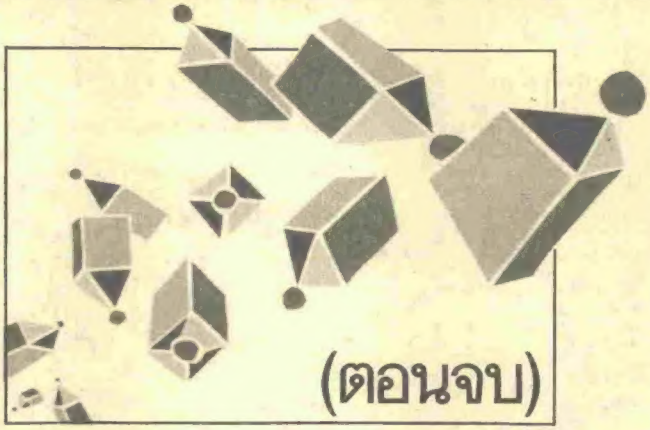
โปรแกรมช่วยศึกษากราฟิกในเทอร์โบปาสคาล



บุญเลิศ เอี่ยมทัศนา

มีผู้ตั้งคำถามผมว่า กราฟิกในเทอร์โบปาสคาลกับในเทอร์โบเบสิกอันไหนเก่งกว่ากัน ในความคิดของผมเห็นว่า กราฟิกในเทอร์โบปาสคาลเน้นหนักไปทางงานด้านธุรกิจ ส่วนกราฟิกในเทอร์โบเบสิกเน้นหนักไปทางด้านการสร้างภาพ

แต่ที่สำคัญคือในเทอร์โบเบสิกทำให้ใช้ World Coordinate ได้คือเมื่อใช้คำสั่ง View ทำให้ขยายและย่อภาพได้ (ดังที่มีใน Turbo Graphics Tool Box ตั้งแต่เทอร์โบปาสคาล รุ่น 3.0) แต่ไม่มีในตัวเทอร์โบปาสคาลรวมทั้งเทอร์โบซีและเทอร์โบโปรล็อกด้วย ผมก็ไม่เข้าใจเหมือนกันว่าทำไมถึงเป็นเช่นนี้ ขอกล่าวเรื่องของผมต่อไปครับ

โปรแกรมช่วย (ต่อ)

ในบทความตอนก่อนผมเว้นการสร้างรูปหลายเหลี่ยมแท้ ๆ ไว้ก่อน เพราะต้องเตรียมโปรแกรมช่วย คือ

SetPoly

ในการวาดรูปหลายเหลี่ยม เราจะต้องกำหนดจุดมุม (vertex) ดังที่เห็นในไมโครฯ 49 รูปที่ 1 และได้กำหนดตัวแปรพอยน์เตอร์ Poly ไว้แล้ว ใช้คู่กับ nump คือจำนวนจุดมุม ซึ่งในโปรแกรมจะดำเนินการตามลำดับดังนี้

- ถ้า Poly ไม่ว่าง ให้ยกเลิกข้อมูลเดิม
- เปิดใช้หน้าต่างบน
- เก็บภาพในหน้าต่างบน แล้วลบหน้าต่างนี้
- ถามผู้ใช้ว่ามีจุดมุมกี่จุด
- ให้ผู้ใช้ดำเนินการป้อนจุดมุมจนครบ (ในการป้อนต้องให้จุดมุมสุดท้ายตรงกับจุดมุมแรก)
- เสร็จแล้วนำภาพในหน้าต่างบนมาคืน เท่ากับเป็นการยืมใช้หน้าต่างบนชั่วคราว
- กลับไปหน้าต่างล่าง

การใช้งานต้องเรียกใช้โปรแกรมนี้ด้วย setpoly ก่อนที่จะใช้คำสั่ง Draw-Poly หรือ FillPoly ดังรูป U2

MakePat

การลงพื้นภาพภายในขอบของรูปวงกลมหรือหลายเหลี่ยม เทอร์โบได้

```
PROCEDURE  SetPoly(var poly : PolyT; var num : Integer);
VAR  p : Pointer;  s,n : Word;  xw, yw : String[30];
     vl : Integer;
BEGIN
  if  poly <> nil  then  FreeMem(poly,num * 4);
  OpenGwindow(top);
  s := ImageSize(0,0, cerwnd.x2, cerwnd.y2);  GetMem(p, s);
  GetImage(0,0, cerwnd.x2, cerwnd.y2,p^);
  ClearViewPort;
  WriteStr('How many vertices? ');
  ReadStream(strm,cx);  GetlValue(strm,cx,num);  nl;
  GetMem(poly,num * 4);
  FOR  n := 1 TO num DO
    begin  xw := 'x' + Chr(n + 48) + ' = ';
           yw := 'y' + Chr(n + 48) + ' = ';  cx := 1;
           WriteStr(xw);  ReadStream(strm,cx);
           GetlValue(strm,cx,vl);
           poly^[n].x := vl;  WriteStr('      ');   cx := 1;
           WriteStr(yw);  ReadStream(strm,cx);
           GetlValue(strm,cx,vl);
           poly^[n].y := vl;  nl;
    end;
  PutImage(0,0,p^,NormalPut);  FreeMem(p, s);  ReadLn;
  OpenGwindow(bot);
END;
```

U2

เตรียมแบบของพื้นภาพไว้ให้แล้ว 11 แบบ และให้เราสร้างได้อีก 1 แบบด้วยคำสั่ง SetFillPattern ซึ่งต้องการข้อมูลที่เป็นแพตเทิร์นขนาด 8 ไบต์ (สำหรับ 8 เส้นสแกนไลน์)

ผมได้เตรียมตัวแปร npat ไว้ให้แล้วและให้โพรซีเดอร์ MakePat ดำเนินการให้โดยที่ข้อมูลที่ต้องการเป็นเลขจำนวนเต็ม Byte จำนวน 8 ไบต์ คือให้ป้อนเป็นเลขฐานสิบหก เพื่อความง่ายในการกำหนดแพตเทิร์น

```
PROCEDURE  MakePat;
var   n : Word;
BEGIN
  if  strm[cx] = #0  then
  begin
  nl;
  WriteStr('Enter a string of 8 two-hexa-digit such as :- ');
  WriteStr('aa AA 31 B2 05 D0 0F F0'); nl;
  ReadStream(strm,cx);
  end;
  for  n := 1 to 8  do
    begin  GetToken(strm,cx,wrd);  HexToNum(wrd,npat[n])  end;
END;
```

U3

```
PROCEDURE  DoDrawPoly(strm : StreamT; cx : Word);
var  num : Integer;
BEGIN
  IF  poly = nil  THEN
    begin  WriteStr('Please  setpoly  first');  Exit  end;
  IF  strm[cx] = #0  THEN  GetParm('(num, Poly)',strm,cx)
  ELSE  OpenGwindow(top);
  GetiValue(strm,cx,num);  DrawPoly(num, poly^);
END;


PROCEDURE  DoFillPoly(strm : StreamT; cx : Word);
var  num : Integer;
BEGIN
  IF  poly = nil  THEN
    begin  WriteStr('Please  setpoly  first');  Exit  end;
  IF  strm[cx] = #0  THEN  GetParm('(num, Poly)',strm,cx)
  ELSE  OpenGwindow(top);
  GetiValue(strm,cx,num);  FillPoly(nump, poly^);
END;
```

P4

การใช้งานให้เรียกใช้ด้วย makepat ก่อนที่จะใช้คำสั่งว่า SetFillPattern ดังแสดงในรูป U3

รูปหลายเหลี่ยม (ต่อ)

การสร้างรูปหลายเหลี่ยมที่ผมจะนำมากล่าวในตอนนี้ คือการสร้างรูปหลายเหลี่ยมด้วย DrawPoly และ FillPoly ดังแสดงในรูป P4 ทั้งคู่เป็นการวาดรูปหลายเหลี่ยม ซึ่งต้องสร้างข้อมูลสำหรับจุดมุมด้วย setpoly ก่อน ข้อแตกต่างกันคือ

- DrawPoly จะวาดเฉพาะขอบของรูปหลายเหลี่ยม ส่วน FillPoly จะลงพื้นภาพให้ด้วย ตามแบบของพื้นภาพที่กำหนดไว้ด้วย SetFillStyle หรือ SetFillPattern

- DrawPoly จะวาดภาพด้วยการต่อจุดมุม ดังนั้นรูปสามเหลี่ยมจะต้องกำหนดจุดมุม 4 จุด ส่วน FillPoly ถือว่าจุดทั้งหมดเป็นจุดมุม ดังนั้นสำหรับรูปสามเหลี่ยม จึงกำหนดจุดมุมเพียง 3 จุด

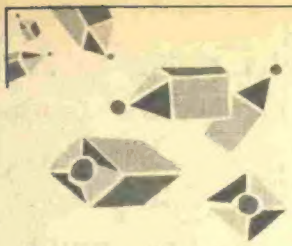
แบบของพื้นภาพและสี

คำในเทอร์โบปาสคาลที่เกี่ยวกับพื้นภาพและสีดังแสดงในรูปที่ 1 ผมได้เว้นการกล่าวถึงคำที่เดียวกับพาเลตของสีทั้งหมด เพราะมุ่งจะใช้โปรแกรมนี้กับจอเฮอร์คิวลิส มีคำที่จะนำมากล่าวถึงคือ

```
        Pattern and Colour
        ==================

/       PROC  SetFillPattern(patarry : FillPatternType; c : Word)
/       PROC  SetFillStyle(pat,c : Word)
x       PROC  GetFillPattern(var pat : FillPatternType);
x       PROC  GetFillSetting(var sett : FillSettingsType);
x       PROC  SetPalette(colnum : Word; c : ShortInt)
x       PROC  SetAllPalette(var cpal)
x       PROC  SetRGBPalette(colnum, redval, greenval, blueval : Integer)
x       FUNC  GetPaletteSize : Word
x       PROC  GetDefaultPalette(var pal : PaletteType)
x       PROC  GetPalette(var pal : PaletteType)
o       PROC  SetBkColor(c : Word)
/       FUNC  GetBkColor : Word;
o       PROC  SetColor(c : Word)
/       FUNC  GetColor : Word;
/       FUNC  GetMaxColor : Word
```

รูปที่ 1
คำในเทอร์โบปาสคาล 5.0
ที่เกี่ยวกับพื้นภาพและสี



DoSetFillPat

เป็นโปรแกรมย่อยสำหรับ SetFillPattern เพื่อจะกำหนดแพตเทิร์นของผู้ใช้ ดังที่กล่าวในหัวข้อ MakePat ในโปรแกรมนี้จึงเตือนให้ผู้ใช้สร้างแพตเทิร์นก่อน ดังแสดงในรูป C1

```
PROCEDURE  DoSetFillPat(strm : StreamT; cx : Word);
var   ch : Char;
BEGIN
  WriteStr('If yet not make user''s patern, press <Esc> ');
  ch := ReadKey;  if  ch = #27  then  Exit;
  IF  strm[cx] = #0  THEN  GetParm('(Pat, c)',strm,cx)
  ELSE  OpenGwindow(top);
  GetToken(strm,cx,wrd);  Get1Value(strm,cx,cc);
  IF  err = 0  THEN  SetFillPattern(npat, cc);
END;


PROCEDURE  DoSetFillStyle(strm : StreamT; cx : Word);
var   pat : Integer;
BEGIN
  IF  strm[cx] = #0  THEN
    GetParm('(pat, c)   pat = 0..12,  12 = user',strm,cx)
  ELSE  OpenGwindow(top);
  Get2Value(strm,cx,pat,cc);
  IF  err = 0  THEN  SetFillStyle(pat, cc);
END;
```

C1

DoSetFillStyle

ในการกำหนดแพตเทิร์นของพื้นภาพนั้น หากต้องการใช้แพตเทิร์นที่เทอร์โบกำหนด สามารถทำได้ด้วยคำสั่ง SetFillStyle ซึ่งมีอยู่ 11 แพตเทิร์น และได้นำแพตเทิร์นของผู้ใช้มาเป็นแพตเทิร์นที่ 12 ดังรูป C1

มีพวกคำเบ็ดเตล็ด คือ GetBkColor, GetColor และ GetMaxColor ซึ่งผมเพียงแต่ให้แสดงค่าออกจอภาพ ดังแสดงในรูป C2

สำหรับคำในกลุ่มนี้ ขอให้สร้างคำต่อไปนี้เพิ่ม คือ DoSetBkColor และ DoSetColor

```
PROCEDURE  DoGetBkColor;
BEGIN  OpenGwindow(top);  Str(GetBkColor,wrd);
       WriteStr(wrd)  END;

PROCEDURE  DoGetColor;
BEGIN  OpenGwindow(top);  Str(GetColor,wrd);
       WriteStr(wrd)  END;

PROCEDURE  DoGetMaxColor;
BEGIN  OpenGwindow(top);  Str(GetMaxColor,wrd);
       WriteStr(wrd)  END;
```

C2

```
      Text
      ====

/     PROC  OutText(astr : String)
/     PROC  OutTextXY(x,y : Integer; astr : String)
/     PROC  SetTextStyle(font, dir csize : Word)
/     PROC  SetTextJustify(hor, ver : Word)
x     PROC  GetTextSetting(var textsett : TextSettingType)
x     PROC  SetUserCharSize(mulX,divX,mulY,divY : Word)
o     FUNC  TextHeight(astr : String): Word
o     FUNC  TextWidth(astr : String) : Word
```

รูปที่ 2 การแสดงภาพอักขระ

ภาพอักขระ

คำในเทอร์โบปาสคาลที่เกี่ยวกับภาพอักขระในกราฟิกโหมด ดังแสดงในรูปที่ 2 มีคำที่ผมจะนำมากล่าวถึงคือ

DoOutText

คำที่ให้นำภาพอักขระไปออกจอภาพคือ OutText ให้นำภาพอักขระไปออกจอภาพโดยถือจุด CP เป็นหลัก กับ OutTextXY ถือจุด (x,y) เป็นหลัก ผมจึงนำมาเขียนรวมกัน ดังนั้นใน DoGraph (รูป G1 ในไมโครฯ 49) ที่เทียบคำทั้งสองนี้ ให้เรียกใช้ DoOutText (wrd, strm, ex) ด้วยกันทั้งคู่ ดังรูป T1

โดยที่ใน OutText จะตรวจดูว่าถ้าเป็น OutTextXY จะต้องการค่า x,y ก่อนสตริง และแยกค่านี้ออกจาก strm ต่อไปเป็นการดำเนินการสำหรับคำทั้งสองนี้ร่วมกัน คือ แยกสตริงใน strm มาให้แก่ astr ตั้งค่า TextJustify และ TextStyle ตามที่เก็บไว้

จากนั้นให้นำออกจอภาพตามแบบของคำสั่งเสร็จแล้วให้ตั้งค่า TextJustify และ TextStyle ให้เป็นไปตามแบบมาตรฐานตามเดิม

DoSetStyle

ภาพอักขระที่จะนำออกจอภาพเราสามารถกำหนด รูปแบบคือเลือกฟอนต์ กำหนดทิศทางในการแสดงคือในทางระดับหรือทางดิ่ง และสามารถกำหนดขนาดของตัวอักษรได้ โดยที่เมื่อ InitGraph เทอร์โบจะกำหนดฟอนต์ไว้เป็นแบบมาตรฐานคือ บิตแม็พให้แสดงในทางระดับ และมีขนาดของตัวอักษรเท่าแบบคือ 1

เราสามารถจะเปลี่ยนแบบของการแสดงได้ด้วยคำสั่ง SetTextStyle ดังรูป T2 ซึ่งเมื่อเปลี่ยนแล้วจะคงอยู่

```
PROCEDURE  DoOutText(comm : WrdT;strm : StreamT;cx : Word);
BEGIN
  IF  (comm = 'OutTextXY')  THEN
    BEGIN
    if  strm[cx] = #0  then
      GetParm('(x,y,string)',strm,cx);
    Get2Value(strm,cx,xx,yy);
    if  err <> 0  then  Exit;
    END;
  IF  cerwnd.name <> top  then  OpenGWindow(top);

  GetString(strm,cx,astr);  SetTextJustify(hor,ver);
  SetTextStyle(font,dir,csize);
  IF  comm = 'OutTextXY'  THEN  OutTextXY(xx,yy,astr)
  ELSE  OutText(astr);
  SetTextStyle(0,0,1);  SetTextJustify(0,2);
END;
```

T1

```
PROCEDURE  DoSetStyle(strm : StreamT;cx : Word);
BEGIN
  IF  strm[cx] = #0  THEN
    begin
    astr := '(font, dir, csize)    ' +
            'font=0..4,  dir=0..1,  csize=1..10';
    GetParm(astr,strm,cx);
    end;
  Get3Value(strm,cx,xx,yy,cc);  if  err <> 0  then  Exit;
  font := xx;  dir := yy;  csize := cc;
END;


PROCEDURE  DoSetJustify(strm : StreamT; cx : Word);
BEGIN
  IF  strm[cx] = #0  THEN
    GetParm('(hor,ver)  hor=0..2,  ver=0..2',strm,cx);
  Get2Value(strm,cx,xx,yy);  if  err <> 0  then  Exit;
  hor := xx;  ver := yy;
END;
```

T2

```
PROCEDURE  DoSetViewPort(strm : StreamT; cx : Word);
var   clip : Boolean;
BEGIN
  IF  strm[cx] = #0  THEN
    GetParm('(x1,y1,x2,y2,clip)   clip = true/false ',strm,cx)
  ELSE  OpenGWindow(top);
  WITH  cerwnd  DO
    BEGIN
    Get4Value(strm,cx,x1,y1,x2,y2);
    GetToken(strm,cx,wrd);    clip := (wrd = 'true');
    IF  err = 0  THEN   SetViewPort(x1,y1,x2,y2,clip);
    END;
END;
```

S1

```
      Screen
      ======

x     PROC  ClearDevice
/     PROC  SetViewPort(x1,y1,x2,y2 : Integer; clip : Boolean)
o     PROC  ClearViewPort
x     PROC  GetViewSetting(var view : ViewPortType)
o     PROC  MoveTo(x,y : Integer)
o     PROC  MoveRel(dx,dy : Integer)
o     FUNC  GetX : Integer
o     FUNC  GetY : Integer
o     FUNC  GetMaxX : Word
o     FUNC  GetMaxY : Word
x     PROC  SetActivePage(pge : Word)
x     PROC  SetVisualPage(pge : Word)
o     PROC  SetWriteMode(wmde : Integer);
x     FUNC  GetMaxMode : Word
x     FUNC  GetModeName(modnum : Word) : String
```

รูปที่ 3 คำในเทอร์โบปาสคาล 5.0 ที่เกี่ยวกับจอภาพ

ตลอดไป จนกว่าจะสั่งเปลี่ยนใหม่ การดำเนินการของ DoSetStyle (ซึ่งในโพรซีเดอร์ DoGraph เมื่อเทียบคำว่า SetTextStyle ให้เรียกใช้โปรแกรมย่อยนี้) จึงเป็นแต่เพียงให้ค่าแก่ตัวแปร font, dir และ csize ใหม่ (การดำเนินการจริงอยู่ใน DoOutText)

DoSetJustify

ตามที่กล่าวใน DoOutText ว่าใช้จุด CP หรือ (x,y) เป็นหลักนั้น ได้ให้ค่าไว้ว่าให้จุดที่เป็นหลักนี้อยู่ที่มุมบนซ้ายของสตริง เราสามารถเปลี่ยนได้ด้วยคำสั่ง SetTextJustify ซึ่งเมื่อเปลี่ยนแล้วจะคงอยู่ตลอดไปเช่นกัน ผมจึงให้ DoSetJustify เปลี่ยนค่า hor และ ver ตามที่ผู้ใช้ป้อน และให้ DoOutText ดำเนินการให้เช่นเดียวกับ DoSetStyle (ในโพรซีเดอร์ DoGraph เมื่อเทียบคำว่า SetTextJustify ให้เรียกใช้โปรแกรมย่อยนี้) ดังรูป T2

จอภาพ

คำที่เกี่ยวกับจอภาพของเทอร์โบปาสคาลดังแสดงในรูปที่ 3 มีคำที่ผมจะนำมากล่าวถึง คือ

DoSetViewPort

เพื่อให้ผู้ใช้ทดลองกำหนดช่องหน้าต่างของตนเองซ้อนกับหน้าต่างบน และเพื่อไม่ให้ซ้อนกับหน้าต่างล่าง ไม่ควรกำหนดค่า Y มากกว่า GetMaxY ลบด้วย 48 โปรแกรมดังรูป S1

และขอให้สร้างโปรแกรมย่อยสำหรับคำต่อไปนี้เพิ่ม คือ ClearViewPort, MoveTo, MoveRel, GetX, GetY, GetMaxX, GetMaxY และ SetWriteMode สำหรับคำนี้ค่า wmde มีค่าเป็น 0 หรือ 1 เท่านั้น ขอให้แสดงคำอธิบายด้วย

การควบคุม

คำที่เกี่ยวกับการดำเนินการควบ

คุม คือ แสดงในรูปที่ 4 คำเหล่านี้ควรอยู่ในการเขียนโปรแกรมมากกว่า ดังที่ปรากฏในโปรแกรมนี้ จึงของดกล่าว

โปรแกรมที่เตรียมไว้สำหรับช่วยการศึกษากราฟิกมีเพียงเท่านี้ ขอให้ป้อนโปรแกรมดังที่กล่าวมาและแทรกคำใน DoGraph ให้สอดคล้องกัน คอมพายล์แล้วทดลองรันดูครับ

เครื่องพิมพ์

โปรแกรมย่อยนี้ไม่เกี่ยวกับการสร้างภาพกราฟิกบนจอภาพ แต่เมื่อผมจะเขียนบทความนี้ต้องการภาพประกอบ ได้ทดลองใช้โปรแกรมอรรถประโยชน์ของการ์ดเฮอร์คิวลิส (hgc) ปรากฏว่ากับเครื่องพิมพ์ของผม Super 5 รุ่น EP-1201 พิมพ์ออกมาทางขวางของกระดาษ และมีขนาดใหญ่เกินความจำเป็น

ผมจึงนำโปรแกรม WaitToGo ของ อ.อาจหาญ สัตยารักษ์ (ไมโครฯ 44) มาดัดแปลงให้ตรงกับความต้องการของผมคือ ให้สามารถพิมพ์ภาพบนจอภาพต่าง ๆ ดังกล่าวมาได้โดยอัตโนมัติ เป็นโปรแกรม PrintGraph รูป B1 ถึง B3 (ขอขอบพระคุณ อ.อาจหาญ สัตยารักษ์ ไว้ ณ ที่นี้ด้วยครับ)

ในการเขียนโปรแกรมนี้ผมพยายามให้เป็นโปรแกรมใช้สอยได้ทั่วไป แต่โปรแกรมที่นำมาแสดงนี้ยังคงอิงอยู่กับยูนิต CRTGP อยู่

โปรแกรมหลัก

โปรแกรมหลักของ PrintGraph รูป B3 เริ่มด้วยการตรวจว่าเครื่องพิมพ์พร้อมหรือไม่ ค่าใน reg.ah ที่นำมาตรวจ ถ้าเป็น 16 แสดงว่าเครื่องพร้อม ถ้าเป็น 8 แสดงว่าเครื่องอยู่ในฐานะ off line เช่นเมื่อกดปุ่ม ON LINE แล้วยังไม่ได้กดซ้ำ ซึ่งจะ on line ให้ทำให้เครื่องอยู่ในฐานะพร้อม ถ้าเครื่องไม่พร้อมจะแจ้งให้ทราบและออกจากโปรแกรมย่อยนี้

จากนี้จะเป็นการส่งโค้ดควบคุมไปยังเครื่องพิมพ์ เพื่อกำหนดค่า line space เป็น 24/180 ของนิ้ว ตั้ง left margin ไว้ 10

โปรแกรมท่อนต่อไปเพื่อเปิดใช้หน้าต่างเต็มจอภาพ อ่านจุดโคออร์ดิเนตสูงสุดของจอภาพมาเก็บไว้ แล้วหาค่า maxStrip ค่านี้ขอเรียนกับผู้ที่ไม่คุ้นเคยกับเครื่องพิมพ์ว่า ในการพิมพ์ภาพกราฟิกตามโปรแกรมนี้จะพิมพ์ครั้งละ 8 เข็มในทางตั้ง (ไม่ว่าจะมีกี่เข็ม) เมื่อให้พิมพ์ต่อเนื่องกันไปก็จะเป็นแถบไปทางขวา (ผมจึงเรียกว่า strip) แถบหนึ่ง ๆ จึงมีความกว้างเท่ากับ 8 เส้นสแกนไลน์ของจอภาพ ดังนั้น จอภาพแต่ละแบบจะมีจำนวนแถบไม่เท่ากัน

การที่ให้เปิดหน้าต่างเต็มจอภาพก็เพื่อที่จะใช้คำสั่ง GetPixel อ่านจุดภาพมาทีละจุด ข้อดีของการใช้คำสั่งนี้ที่เหนือกว่าที่เราจะอ่านโดยตรงจากหน่วยความจำจอภาพ คือสามารถใช้กับจอภาพได้ทุกแบบโดยไม่ต้องตั้งเงื่อนไขให้ยุ่งยาก และผมคิดว่าหากดำเนินการเองคงไม่เร็วกว่าใช้คำสั่งนี้เท่าไร

จากนั้นเป็นการตรวจว่าเป็นจอภาพแบบใด ซึ่งตรวจละเอียดเฉพาะจอสีเท่านั้น เพื่อเลือกโหมดการพิมพ์ให้สอดคล้องกัน แล้วดำเนินการพิมพ์ครั้งละ 1 แถบจนสุดจอภาพ แล้วปรับ

```
Control
=======

PROC  InitGraph(var Gd,Gm : Integer; drvpath : String)
PROC  SetGraphMode(mde : Integer)   {*}
FUNC  GetGraphMode : Integer;   {*}
PROC  RestoreCrtMode
PROC  DetectGraph(var Gd,Gm : Integer)
PROC  CloseGraph
PROC  GraphDefaults
FUNC  GetMaxMode : Word
FUNC  GetModeName(modnum : Word) : String
PROC  GetModeRange(Gd : Integer; var Lom,Him : Integer)
PROC  SetGraphBufSize(bufsize : Word)
FUNC  GetDriverName : String;
FUNC  GraphResult : Integer
FUNC  GraphErrorMsg(errcode : Integer) : String
FUNC  InstallUserDriver(nme : String; autodet : Pointer) : Word
FUNC  InstallUserFont(fontfi : String) : Integer
FUNC  RegisterBGIdriver(drv : Pointer) : Integer
FUNC  RegisterBGIfont(font : Pointer) : Integer
```

รูปที่ 4 คำในเทอร์โบปาสคาล 5.0 ที่เกี่ยวกับการควบคุม

```
BEGIN  { PrintGraph }
  reg.ah := 1;  reg.dx := 0;  Intr($17, reg);
  IF  (reg.ah <> 16 ) AND (reg.ah <> 8)  THEN
    begin  WriteStr('  PRINTER NOT READY  try again');
           Write(^G);   Exit  end;

  SentControl( #27 + '3' + ASPECT); { Set line space }
  SentControl( #27 + 'l' + #10);    { Set Left Margin }

  oldwnd := cerwnd.name;  OpenGWindow(main);
  maxX := GetMaxX;  maxY := GetMaxY;
  maxStrip := maxY DIV 8;
  DetectGraph(Gd,Gm);
  IF  (Gd = CGA) AND (Gm in [0..3])
    THEN  prMode := 0      { 480 dot/line }
  ELSE  prMode := 1;       { 960 dot/line }
  n1 := Lo(Succ(maxX));  n2 := Hi(Succ(maxX));

  FOR  lne := 0 TO maxStrip  DO  PrintStrip(lne);
  SentControl(#27 + '2');
  SentControl(#27 + 'l' + #0);  SentControl(#13);

  OpenGWindow(oldwnd);
END;
```

B3

เครื่องพิมพ์ให้กลับสู่สถานะเดิม

PrintStrip

โปรแกรมนี้จะพิมพ์จุดภาพบนจอภาพ 1 แถบ เริ่มโปรแกรมด้วยการตั้งโหมดการพิมพ์ด้วย SentControl ตามที่เตรียมค่าไว้ในโปรแกรมหลัก ที่ต้องตั้งทุกครั้งเพราะจะถูกยกเลิกเมื่อให้ขึ้นบรรทัดใหม่ (from feed) คือ เมื่อพิมพ์ข้อมูลครบ 1 แถบ แล้วเราต้องให้ขึ้นบรรทัดใหม่ เพื่อเป็นการเริ่มต้นแถบใหม่ ดังแสดงในรูป B2

ในการส่งข้อมูลออกจอภาพนั้น ดังที่กล่าวแล้วว่า หัวพิมพ์จะพิมพ์ภาพในทางตั้ง คือแกน y ครั้งละ 1 ไบต์ แต่การเก็บภาพในหน่วยความจำเก็บเรียงในทางนอนคือ แกน x

ดังนั้นจึงต้องอ่านค่าในแกน y ตรงคอลัมน์ xx มาครั้งละ 1 บิต นำมาหาน้ำหนักตามที่กำหนดในคู่มือเครื่องพิมพ์จนครบ 8 บิต ดังค่าที่ให้แก่ตัวแปร bit ได้เป็นข้อมูล 1 ไบต์ส่งไปให้เครื่องพิมพ์ด้วยอินเตอร์รัพต์ $17 โดยตรง ไม่ผ่านโปรแกรมย่อย เพื่อให้ดำเนินการได้เร็วขึ้น

ข้อมูลที่ส่งไปให้ครั้งละ 1 ไบต์นี้ เครื่องพิมพ์จะนำไปเก็บไว้ในบัฟเฟอร์ของตน จนเมื่อเราอ่านภาพจนสุดแถบ คือตั้งแต่ xx เท่ากับ 0 จนถึง maxX ได้ให้รหัส from feed คือที่ Out1Byte ด้วยค่า 10 เครื่องพิมพ์จึงจะดำเนินการพิมพ์ แล้วขึ้นบรรทัดคือแถบใหม่

มีโปรแกรมสนับสนุนการทำงานของ PrintGraph อีก 2 โปรแกรม ดังแสดงในรูป B1 คือ

Out1Byte

เพื่อส่งข้อมูลหรือรหัส 1 ไบต์ไปให้แก่เครื่องพิมพ์ แต่ในวงรอบ FOR ของ PrintStrip ไม่ได้เรียกใช้โปรแกรมย่อยนี้ เพราะต้องการประหยัดเวลาดังกล่าวแล้ว

SentControl

เพื่อส่งข้อมูลหรือรหัสหลาย ๆ ไบต์ ไปให้แก่เครื่องพิมพ์ ซึ่งในโปรแกรมนี้ใช้ส่งรหัสเท่านั้น

ขอให้ป้อนโปรแกรมนี้ในไฟล์ CRTGP.PAS ตามที่กำหนดไว้ แล้วคอมพายล์ลงดิสค์ การใช้งานเมื่อต้องการพิมพ์ภาพบนจอภาพให้กดคีย์ Ctrl-P หรือจะใช้คำสั่ง prng ก็ได้ ซึ่งก่อนที่จะพิมพ์บรรทัดแรกจะต้องรอนานพอสมควร (สำหรับเครื่องที่ความเร็วสัญญาณนาฬิกา 4.77 MHz)

```
PROCEDURE  PrintGraph;
CONST    ASPECT = #24;
TYPE     StrokeT = String[7];
VAR      prmode, maxX, maxY, maxStrip,
                              n1, n2, lne : Integer;
         Gd, Gm : Integer;    oldwnd : WndName;
         reg : Registers;


procedure  Out1Byte(dat : Byte);
var      reg : Registers;
begin
  reg.ah := 0;  reg.al := dat;
  reg.dx := $0000;  Intr($17, reg);
end;

procedure  SentControl(stroke : StrokeT);
var    id : Byte;
begin
  for  id := 1 to Length(stroke)  do
    Out1Byte(Ord(stroke[id]));
end;
```

B1

```
procedure  PrintStrip(lne : Integer);
const  Bits : ARRAY[0..7] OF Byte =
                (128,64,32,16,8,4,2,1);
var      bit, dat : Byte;
         xx, yy : Integer;
begin
  yy := lne * 8;
  SentControl( #27 + '*' + Chr(prMode) +
               Chr(n1) + Chr(n2) + #0);
  for  xx := 0 to maxX  do
    begin
    dat := 0;
    for  bit := 0 to 7  do
      if  GetPixel(xx,yy + bit) <> 0  then
        dat := dat OR Bits[bit];
    reg.ah := 0;  reg.dx := $0000;
    reg.al := dat;  Intr($17, reg);
    end;
  Out1Byte(10);
end;
```

B2

สร้างคำเพิ่ม

นอกจากคำในเทอร์โบปาสคาลแล้ว ผมเห็นว่าควรสร้างคำใหม่เพิ่มเพื่ออำนวยความสะดวกในการวาดภาพ คือ

Draw

ในเทอร์โบเบสิก มีคำว่า Draw เพื่อใช้สร้างภาพตามคำที่กำหนดในสตริงในรูปแบบของเวกเตอร์ เช่น ถ้าเขียนด้วยภาษาปาสคาลจะเขียนได้เป็น

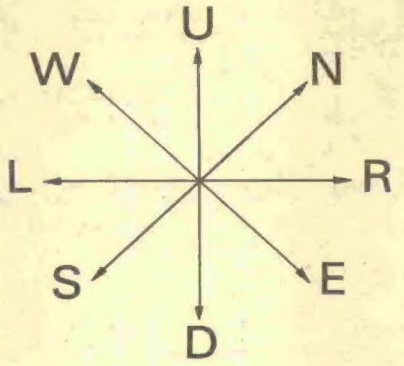
```
Draw(u20r20d20l20);
```

หมายถึงให้ลากเส้นขึ้นบนด้วยความยาว 20 จุด แล้วลากไปทางขวา 20 จุด ลงล่าง 20 จุด ไปทางซ้าย 20 จุด เกิดเป็นภาพรูปสี่เหลี่ยม ทำให้วาดภาพได้โดยสะดวก และรวดเร็วพอสมควร แต่ไม่มีในเทอร์โบปาสคาล ผมจึงสร้างคำนี้ขึ้นและขอนำมาเสนอ

โดยได้กำหนดความหมายของอักษรควบคุมบางตัวใหม่ให้สอดคล้องกับภาษาปาสคาลดังแสดงในรูปที่ 5 อักษรควบคุมเหล่านี้บางตัวต้องการค่า 1 ค่า บางตัวต้องการ 2 ค่า ซึ่งจะต้องแยกจากกันด้วยเครื่องหมาย , (ขอให้ดูตัวอย่างได้ในรูป E4 คือค่าที่ให้แก่ตัวแปรสตริง sail ซึ่งเป็นรูปเรือใบดังผลการรันในรูปที่ 6)

นำไปเขียนโปรแกรมเป็นโพรซีเดอร์ DrawVect ดังรูป E1 และ E2 โดยที่โปรแกรมนี้จะอ่านอักษรควบคุมในสตริง astr มาตีความทีละอักษร และอ่านแถวของตัวเลขมาแปลงเป็นเลขจำนวน 1 หรือ 2 จำนวนดังกล่าวแล้วแล้วลากเส้นตามที่กำหนดด้วยอักษรควบคุม

การแปลงแถวของตัวเลขเป็นเลขจำนวนโพรซีเดอร์ Value จะดำเนินการครั้งละ 1 จำนวน และจะตรวจว่ามีเครื่องหมาย , หลังตัวเลขหรือไม่ ถ้ามีจะเลื่อนอินเด็กซ์ข้ามเครื่องหมายนี้ไว้ให้เพื่อการดำเนินการในครั้งต่อไป นอกจากนี้ยังให้สามารถขยายภาพได้ด้วย



- อักษร U, R, D, L, N, E, S และ W ให้ลากเส้นในทิศทางตามภาพยาว x จุด
- T ให้ลากเส้นไปยังจุด x,y ที่สัมพันธ์กับจุด CP
- M ให้เลื่อนจุด CP ไปยังจุด x,y ที่สัมพันธ์กับจุด CP
- C ถ้าเป็นอักษรตัวแรก x คือค่าสี ถ้าเป็นอักษรตัวต่อ ๆ ไปให้วาดวงกลมด้วยรัศมี x

หมายเหตุ ค่า x หรือ x, y คือ ตัวเลขที่ตามมา

รูปที่ 5 ความหมายของอักษรควบคุม

```
PROCEDURE  DrawVect(astr : String; si : Word);
VAR   id,xx,yy,nn,oc,cc : Integer;  ch,sign : Char;

procedure  Value(var id,num : Integer);
var    st : String[5];  res : Integer;
begin
  if  astr[id] = '-'  then
    begin  inc(id);  sign := '-'  end
  else  sign := '+';
  st := '';
  while  astr[id] in ['0'..'9']  do
    begin   st := st + astr[id];  inc(id)  end;
  if  astr[id] = ','  then  inc(id);
  Val(st,num,res);
  num := num * si;
  if  sign = '-'  then  num := -num;
end;
```

E1

```
BEGIN
  oc := GetColor;
  for  id := 1 to Length(astr)
    do  astr[id] := UpCase(astr[id]);
  id := 1;
  IF  astr[id] = 'C' THEN
    begin  inc(id);  Value(id,cc);
           SetColor(cc)  end;

  WHILE  id < Length(astr)  DO
    BEGIN
    ch := astr[id];  inc(id);  Value(id,nn);
    IF  ch = 'M'  THEN
      begin  Value(id,yy);  MoveRel(nn,yy)  end
    ELSE
      BEGIN
      CASE  ch  OF
        'U' : begin  xx := 0;    yy := -nn  end;
        'R' : begin  xx := nn;   yy := 0    end;
        'D' : begin  xx := 0;    yy :=  nn  end;
        'L' : begin  xx := -nn;  yy := 0    end;
        'N' : begin  xx := nn;   yy := -nn  end;
        'E' : begin  xx := nn;   yy := nn   end;
        'S' : begin  xx := -nn;  yy := nn   end;
        'W' : begin  xx := -nn;  yy := -nn  end;
        'T' : begin  xx := nn;  Value(id,yy)  end;
        'C' : begin  Circle(GetX,GetY,nn);  Exit  end;
      END;  { case }
      LineRel(xx,yy);
      END;
    END;

    SetColor(oc);
END;
```

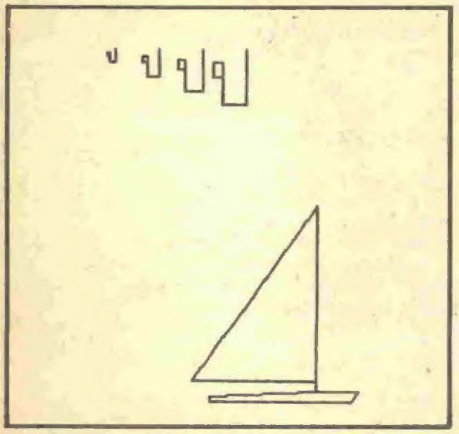
E2

โดยนำค่าที่หามาได้มาคูณด้วยค่า si ซึ่งหมายถึงขนาด

ในการสร้างภาพควรพล็อตลงบนกระดาษกราฟก่อน ส่วนการป้อนสตริงที่จะเป็นภาพให้ป้อนติดกันโดยไม่เว้นวรรค และใช้เครื่องหมาย , เพื่อแยกเลข 2 จำนวนเท่านั้น หากสตริงยาวเกินบรรทัดให้ใช้เครื่องหมายคำพูดเดียวปิดตามด้วย + (คือ Concat) แล้วเปิดเครื่องหมายคำพูดใหม่ แต่เมื่อรวมกันแล้วต้องยาวไม่เกิน 255 อักขระ

โปรแกรมนี้สามารถปรับปรุงได้อีก เช่น ให้สามารถลากเส้นแบบต่าง ๆ (เช่น เส้นประ) ได้ แต่ผมเห็นว่าการลากเส้นแบบนี้ เหมาะสำหรับการวาดภาพมากกว่าตัวอักษร ซึ่งเป็นการลากเส้นสั้น ๆ ที่ต้องการความรวดเร็วมากกว่า จึงเขียนโพรซีเดอร์ DrawCh รูป E3 เพื่อการนี้

โดยกำหนดแบบของสตริงที่จะใช้ใหม่ ดังที่ให้แก่ตัวแปร goh และ poh (หมายถึง ก และ ป) รูป E4 คือตัว



รูปที่ 6 ภาพที่วาดด้วย DrawCh และ DrawVector

```
PROCEDURE  DrawCh(astr : String; si : Word);
VAR   id,xx,yy,nn,oc,cc : Integer;  ch : Char;


procedure  Value(var id,num : Integer);
var    res : Integer;
begin
  if  astr[id] = '-'  then
    begin  inc(id);  num := Ord(astr[id]) - 48;
           num := -num  end
  else  num := Ord(astr[id]) - 48;
  num := num * si;
  inc(id);
end;
```

E3

```
PROGRAM  TestDraw;
USES   crt,graph,crtgp;
TYPE   FinameT = String[79];
CONST  goh : String =  'm28u3n1w1n1r4e1d5m3-8';
       poh : String =  'm24L2u2r2d6r5u8m20';
       ring : String = 'm99c5';
       sail : String = 'm6,50r40n2L12t-30,1d1m30,-2u35t-35,33r35';
VAR    id  : Integer;

{$I DRAW.ICL}

begin   { main }
  InitWindow;  OpenGWindow(main);

  for  id := 1  to 4  do
    begin  MoveTo(id*30,0);  DrawCh(poh,id)  end;

  MoveTo(100,50);  DrawVect(sail,3);

ReadLn;  CloseGraph;
end.
```

E4

```
PROCEDURE  LineAngDist(ang,dist : Word);
VAR  rad : Real;  x,y : Integer;
BEGIN
  rad := (ang + 90) * Pi / 180;
  x := Round(Sin(rad) * dist);
  y := Round(Cos(rad) * dist);
  LineRel(x,y);
END;
```

D

เลขต้องเป็นหลักเดียว และการเขียนเลข 2 จำนวนให้เขียนติดกัน โดยไม่ต้องมีเครื่องหมาย , คั่น

ข้อแตกต่างของโพรซีเดอร์ DrawCh กับ DrawVect อยู่ตรงโปรแกรมย่อย Value ส่วนโปรแกรมหลักของมันคงเหมือนกัน (จึงไม่ได้แสดงในรูป E3) ขอให้ป้อนโพรซีเดอร์ทั้งสองนี้ลงไฟล์ DRAW.ICL เพื่อเก็บไว้ใช้งานต่อไป ได้เขียนโปรแกรมเพื่อทดลองใช้คำทั้งสองนี้ในรูป E4 ได้ผลการรันดังรูปที่ 6

ที่ผมกล่าวว่าการแสดงภาพแบบนี้เป็นแบบเวกเตอร์นั้นไม่ถูกต้องนัก เพราะข้อมูลแบบเวกเตอร์จริง ๆ นั้น ต้องใช้ในการหมุน หันภาพได้ด้วย (ดูบทความเกี่ยวกับเรื่องกราฟิกในไมโครฯ 43)

LineAngDist

บ่อยครั้งที่เราต้องการลากเส้นโดยอ้างถึงมุมกับระยะ เช่น ลากเส้น (30, 100) หมายถึง ให้ลากเส้นจากจุด CP ทำมุม 30 องศากับแกน x ไปเป็นระยะทาง 100 จุด ผมจึงเขียนโพรซีเดอร์ LineAngDist ดังแสดงในรูป D เพื่อการนี้หวังว่าคงไม่ต้องการคำอธิบายใด

พบกันใหม่ฉบับหน้าในเรื่อง "ทำภาษาไทยในกราฟิกโหมด" ซึ่งหมายถึงการแสดงภาพในแบบบิตแม็พด้วยครับ